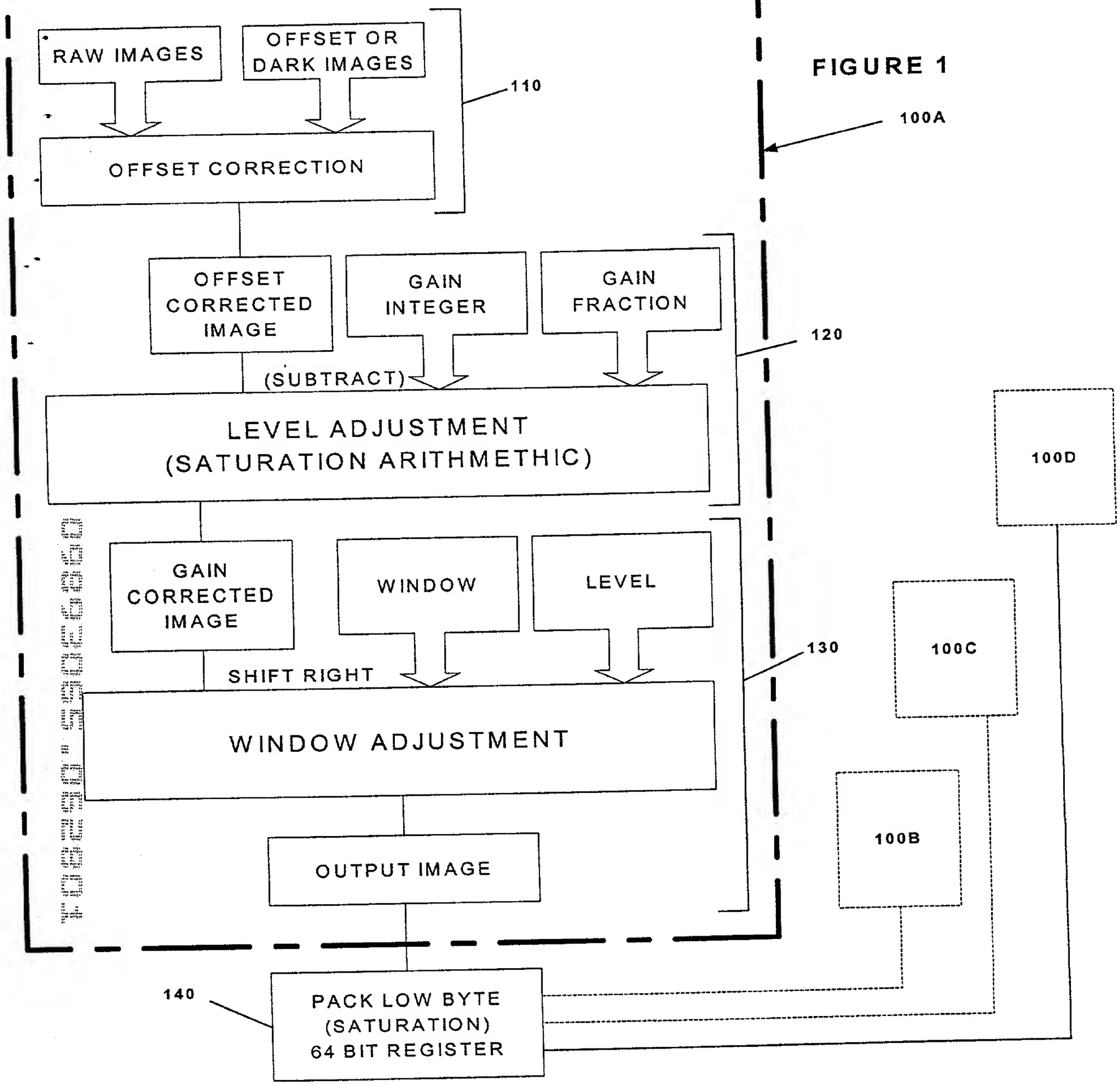

FIGURE 1

FIGURE 1

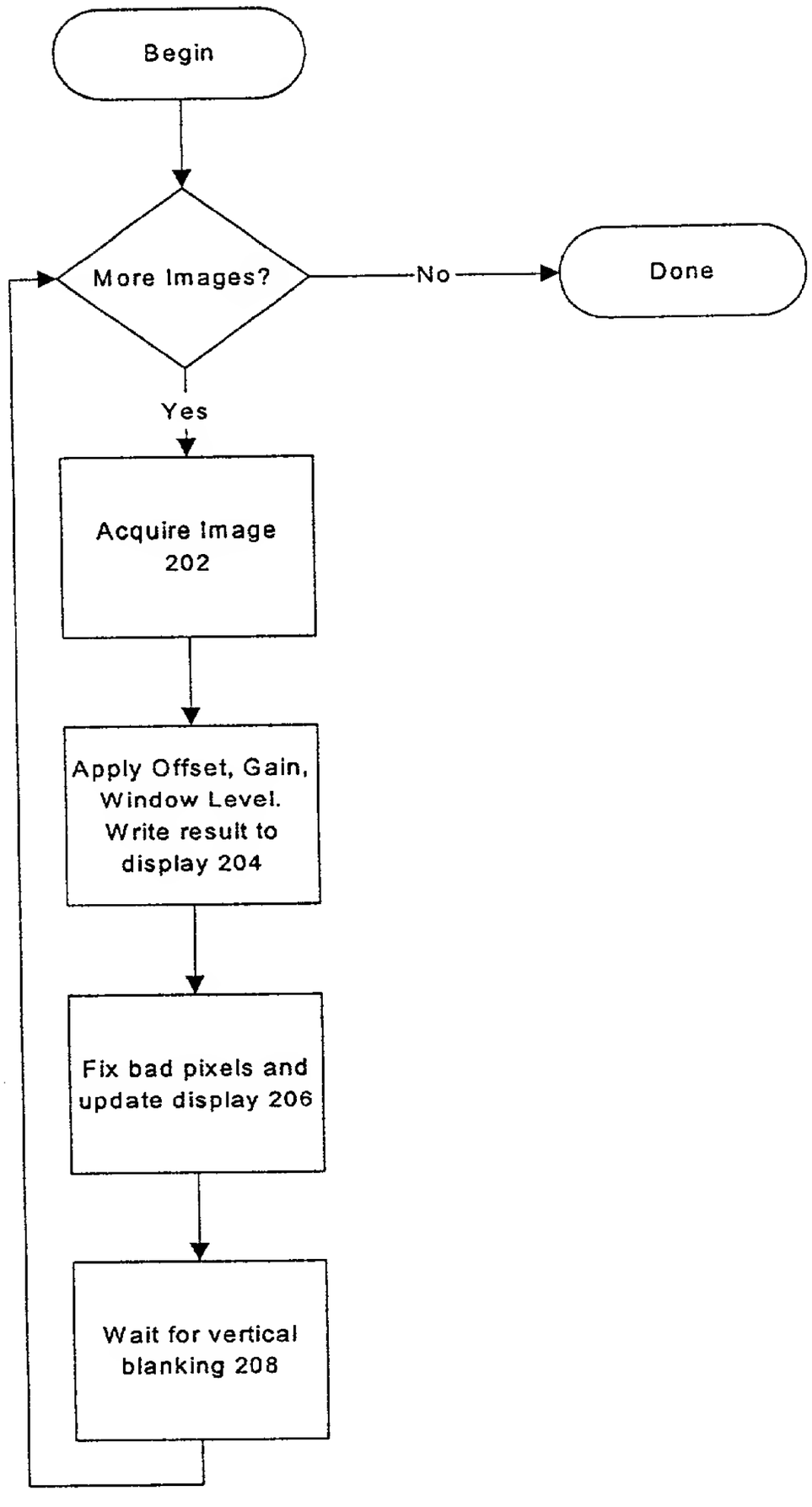
Begin
More Images?
No
Done
Yes
Acquire Image 202
Apply Offset, Gain, Window Level. Write result to display 204
Fix bad pixels and update display 206
Wait for vertical blanking 208

FIGURE 2

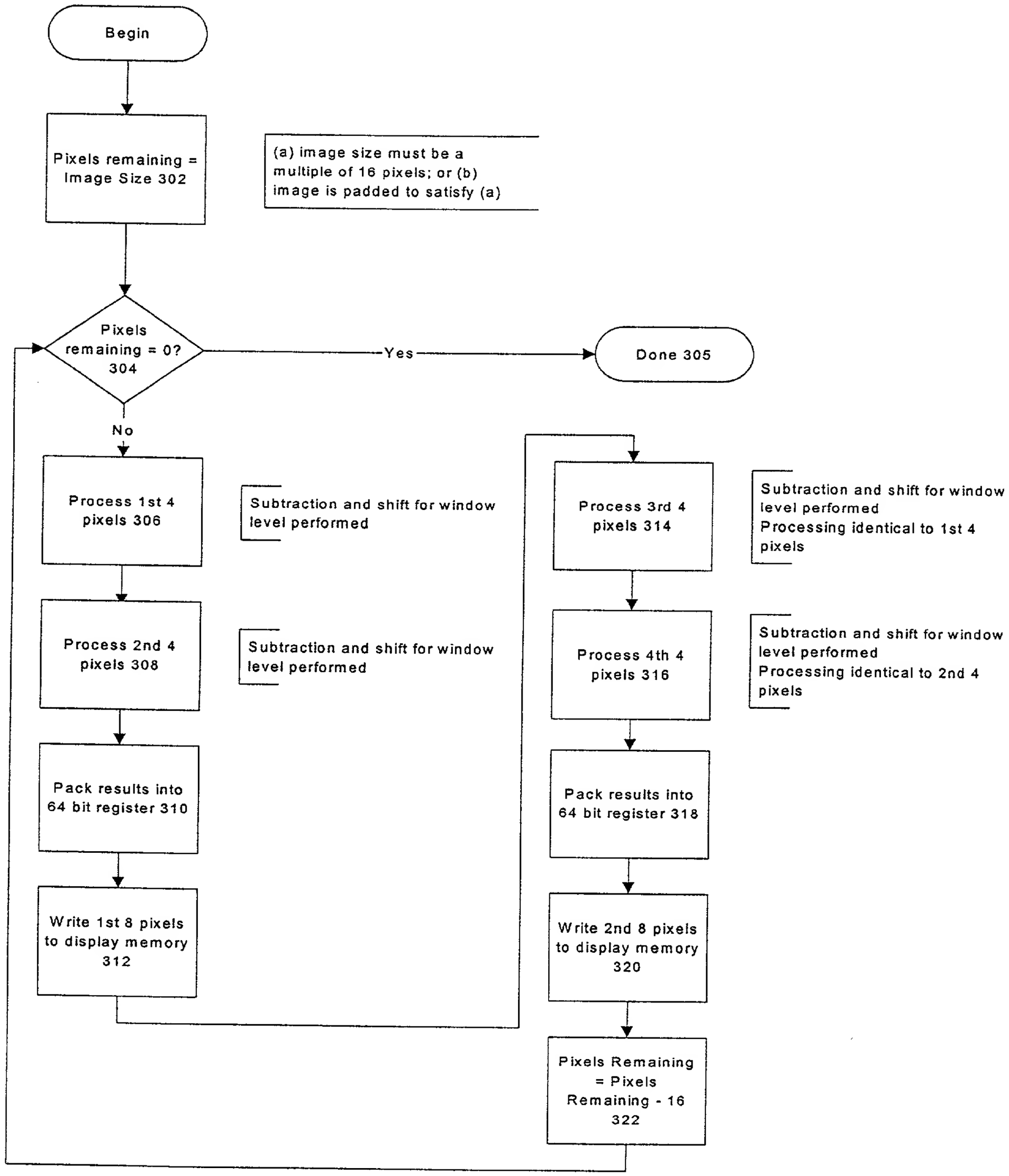
Begin
Pixels remaining = Image Size 302
(a) image size must be a multiple of 16 pixels; or (b) image is padded to satisfy (a)
Pixels remaining = 0? 304
Yes
Done 305
No
Process 1st 4 pixels 306
Subtraction and shift for window level performed
Process 2nd 4 pixels 308
Subtraction and shift for window level performed
Pack results into 64 bit register 310
Write 1st 8 pixels to display memory 312
Process 3rd 4 pixels 314
Subtraction and shift for window level performed
Processing identical to 1st 4 pixels
Process 4th 4 pixels 316
Subtraction and shift for window level performed
Processing identical to 2nd 4 pixels
Pack results into 64 bit register 318
Write 2nd 8 pixels to display memory 320
Pixels Remaining = Pixels Remaining - 16 322

FIGURE 3

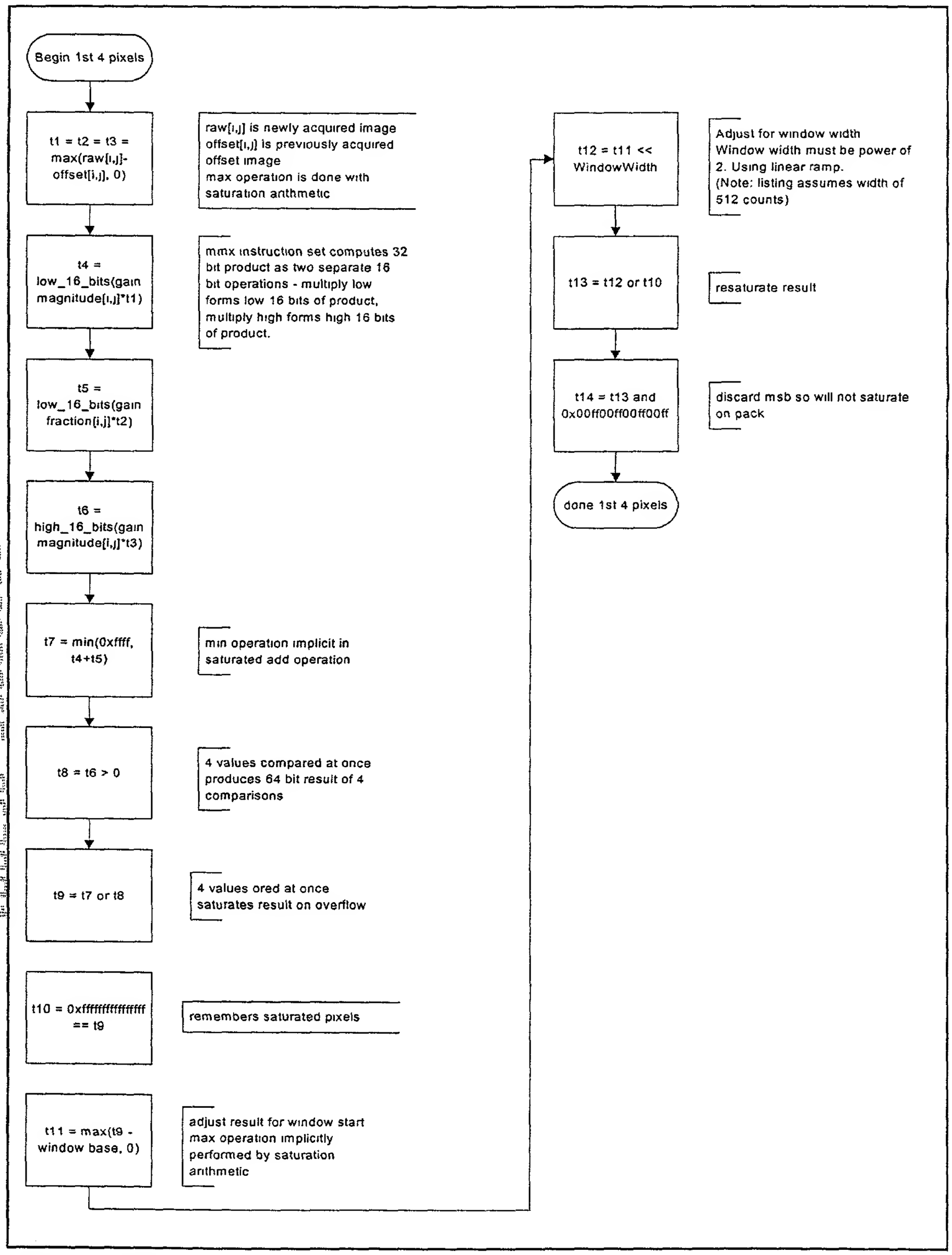
Begin 1st 4 pixels
t1 = t2 = t3 = max(raw[i,j]-offset[i,j], 0)
raw[i,j] is newly acquired image
offset[i,j] is previously acquired offset image
max operation is done with saturation arithmetic
t4 = low_16_bits(gain magnitude[i,j]*t1)
mmx instruction set computes 32 bit product as two separate 16 bit operations - multiply low forms low 16 bits of product, multiply high forms high 16 bits of product.
t5 = low_16_bits(gain fraction[i,j]*t2)
t6 = high_16_bits(gain magnitude[i,j]*t3)
t7 = min(0xffff, t4+t5)
min operation implicit in saturated add operation
t8 = t6 > 0
4 values compared at once produces 64 bit result of 4 comparisons
t9 = t7 or t8
4 values ored at once saturates result on overflow
t10 = 0xffffffffffffffff == t9
remembers saturated pixels
t11 = max(t9 - window base, 0)
adjust result for window start
max operation implicitly performed by saturation arithmetic
t12 = t11 << WindowWidth
Adjust for window width
Window width must be power of 2. Using linear ramp.
(Note: listing assumes width of 512 counts)
t13 = t12 or t10
resaturate result
t14 = t13 and 0x00ff00ff00ff00ff
discard msb so will not saturate on pack
done 1st 4 pixels

FIGURE 4

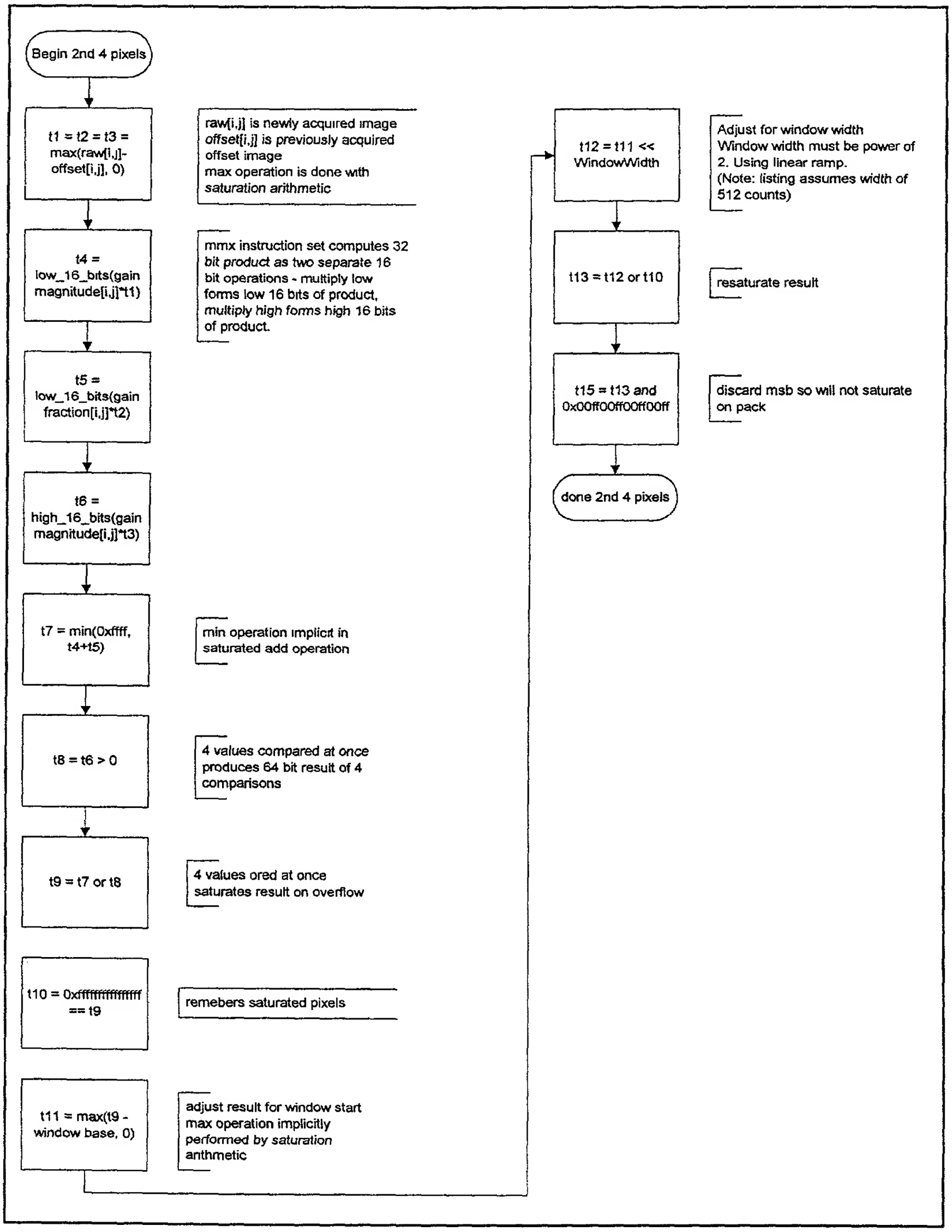
Begin 2nd 4 pixels
t1 = t2 = t3 = max(raw[i,j]-offset[i,j], 0)
raw[i,j] is newly acquired image
offset[i,j] is previously acquired offset image
max operation is done with saturation arithmetic
t4 = low_16_bits(gain magnitude[i,j]*t1)
mmx instruction set computes 32 bit product as two separate 16 bit operations - multiply low forms low 16 bits of product, multiply high forms high 16 bits of product.
t5 = low_16_bits(gain fraction[i,j]*t2)
t6 = high_16_bits(gain magnitude[i,j]*t3)
t7 = min(0xffff, t4+t5)
min operation implicit in saturated add operation
t8 = t6 > 0
4 values compared at once produces 64 bit result of 4 comparisons
t9 = t7 or t8
4 values ored at once saturates result on overflow
t10 = 0xffffffffffffffff == t9
remebers saturated pixels
t11 = max(t9 - window base, 0)
adjust result for window start
max operation implicitly performed by saturation arithmetic
t12 = t11 << WindowWidth
Adjust for window width
Window width must be power of 2. Using linear ramp.
(Note: listing assumes width of 512 counts)
t13 = t12 or t10
resaturate result
t15 = t13 and 0x00ff00ff00ff00ff
discard msb so will not saturate on pack
done 2nd 4 pixels

FIGURE 5

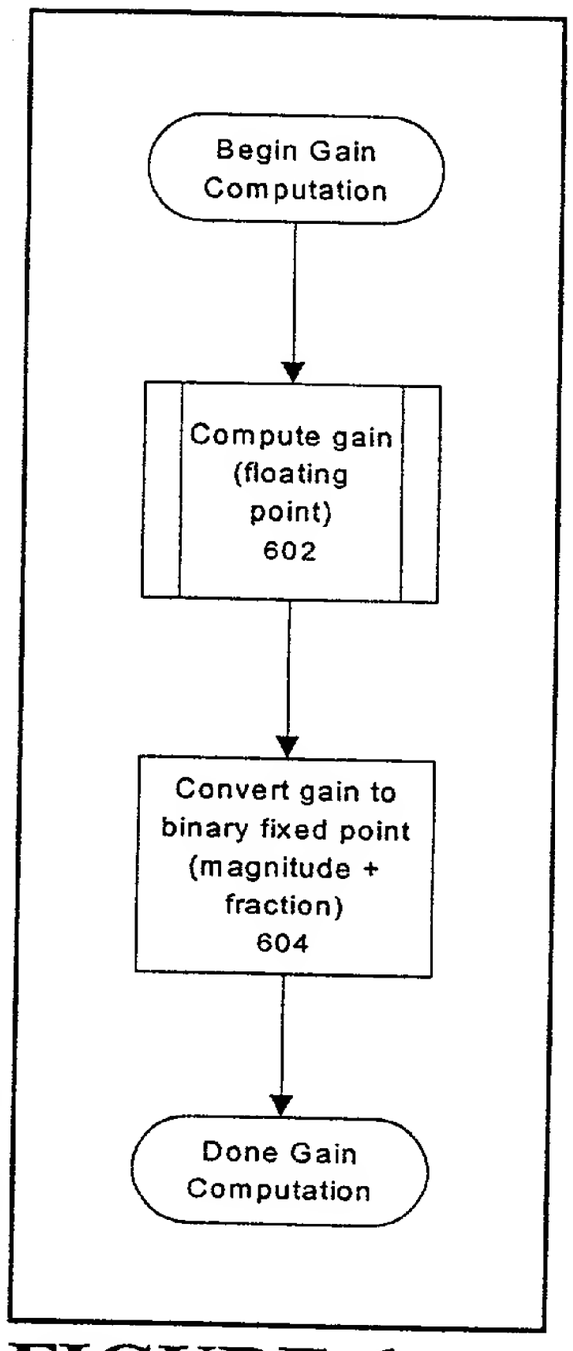

FIGURE 6

Do not copy, post, or distribute

7/13

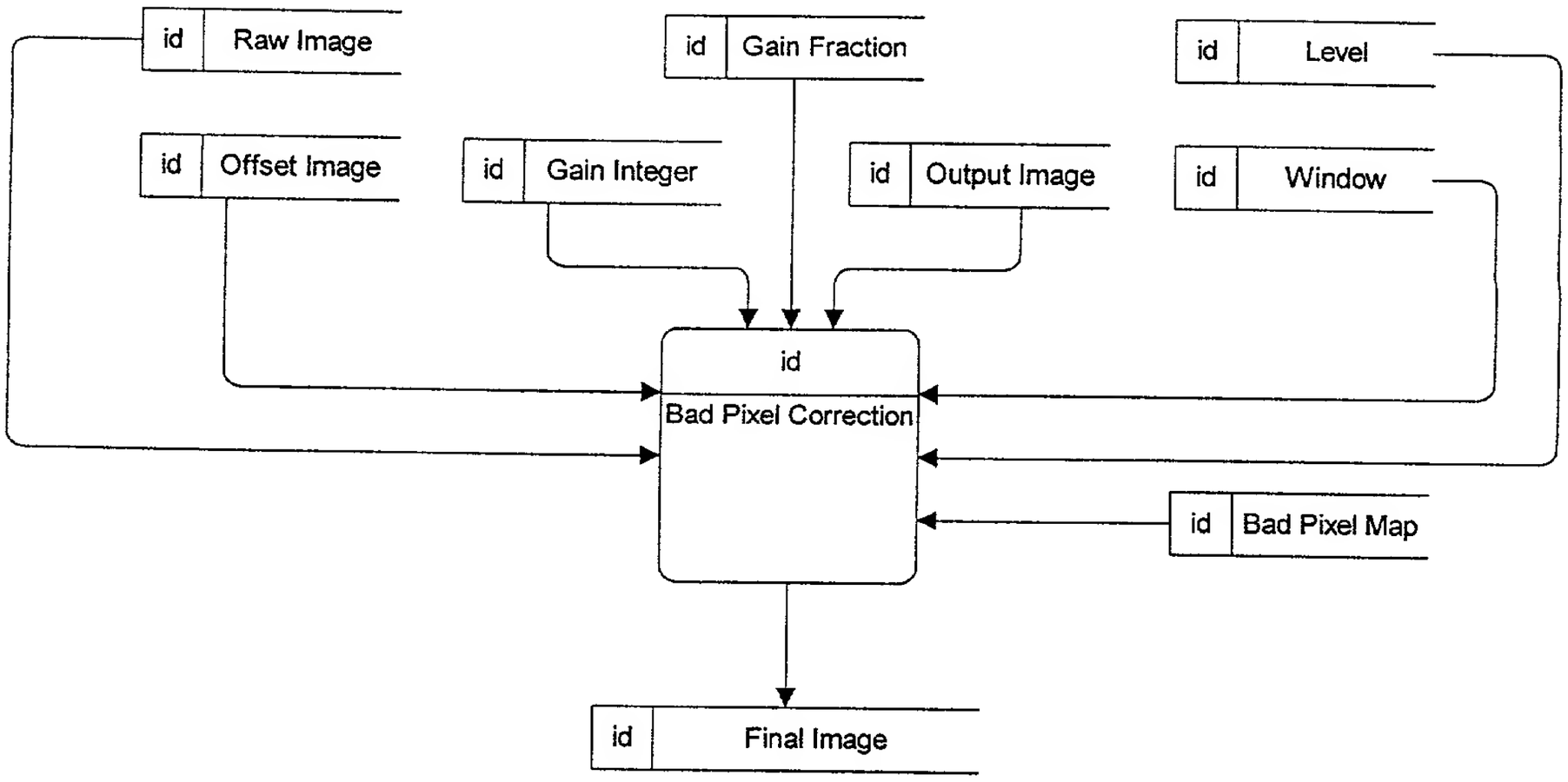
id
Raw Image
id
Gain Fraction
id
Level
id
Offset Image
id
Gain Integer
id
Output Image
id
Window
id
Bad Pixel Correction
id
Bad Pixel Map
id
Final Image

FIGURE 7

8/13

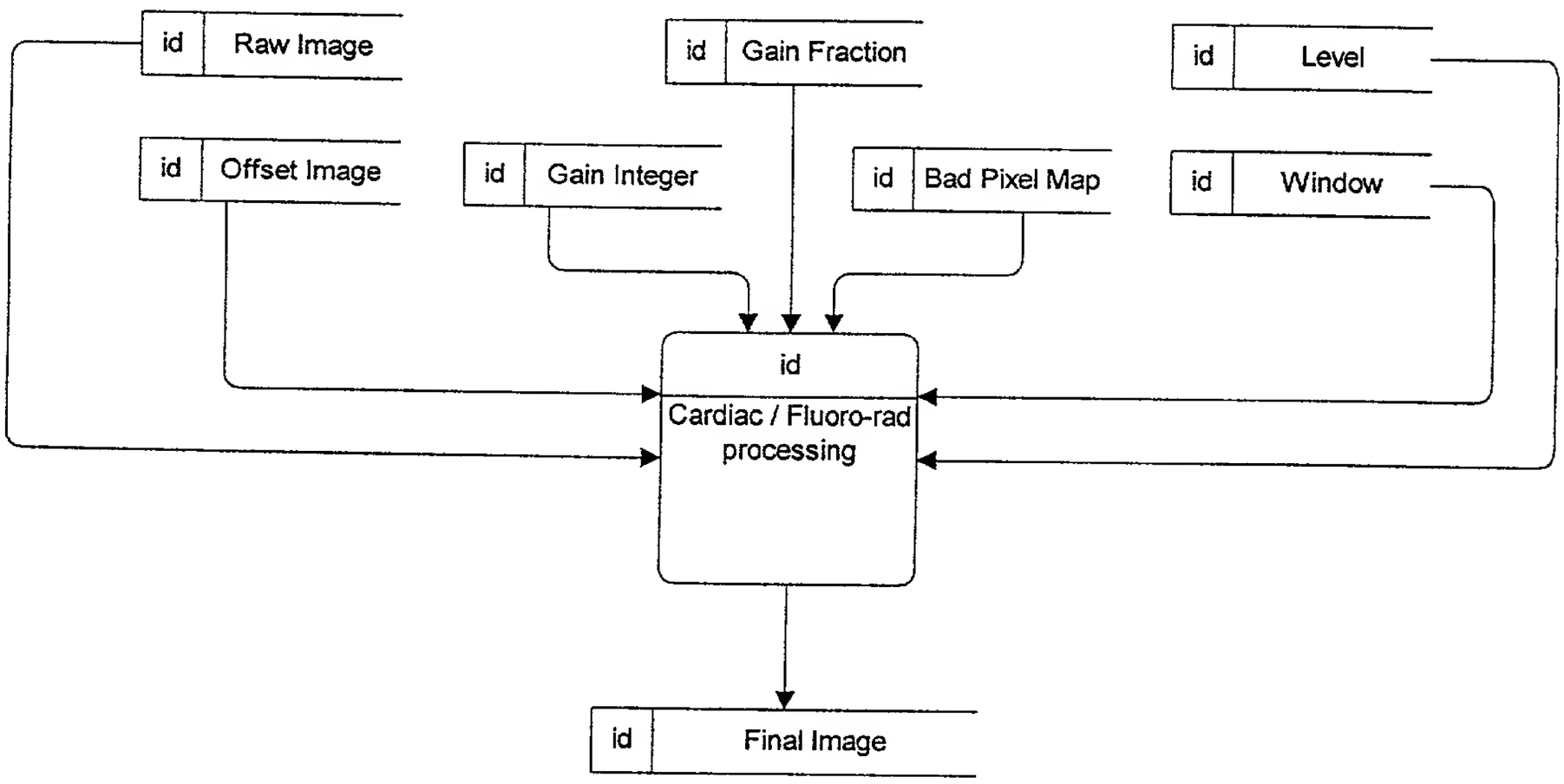
id
Raw Image
id
Gain Fraction
id
Level
id
Offset Image
id
Gain Integer
id
Bad Pixel Map
id
Window
id
Cardiac / Fluoro-rad processing
id
Final Image

FIGURE 8

9/13

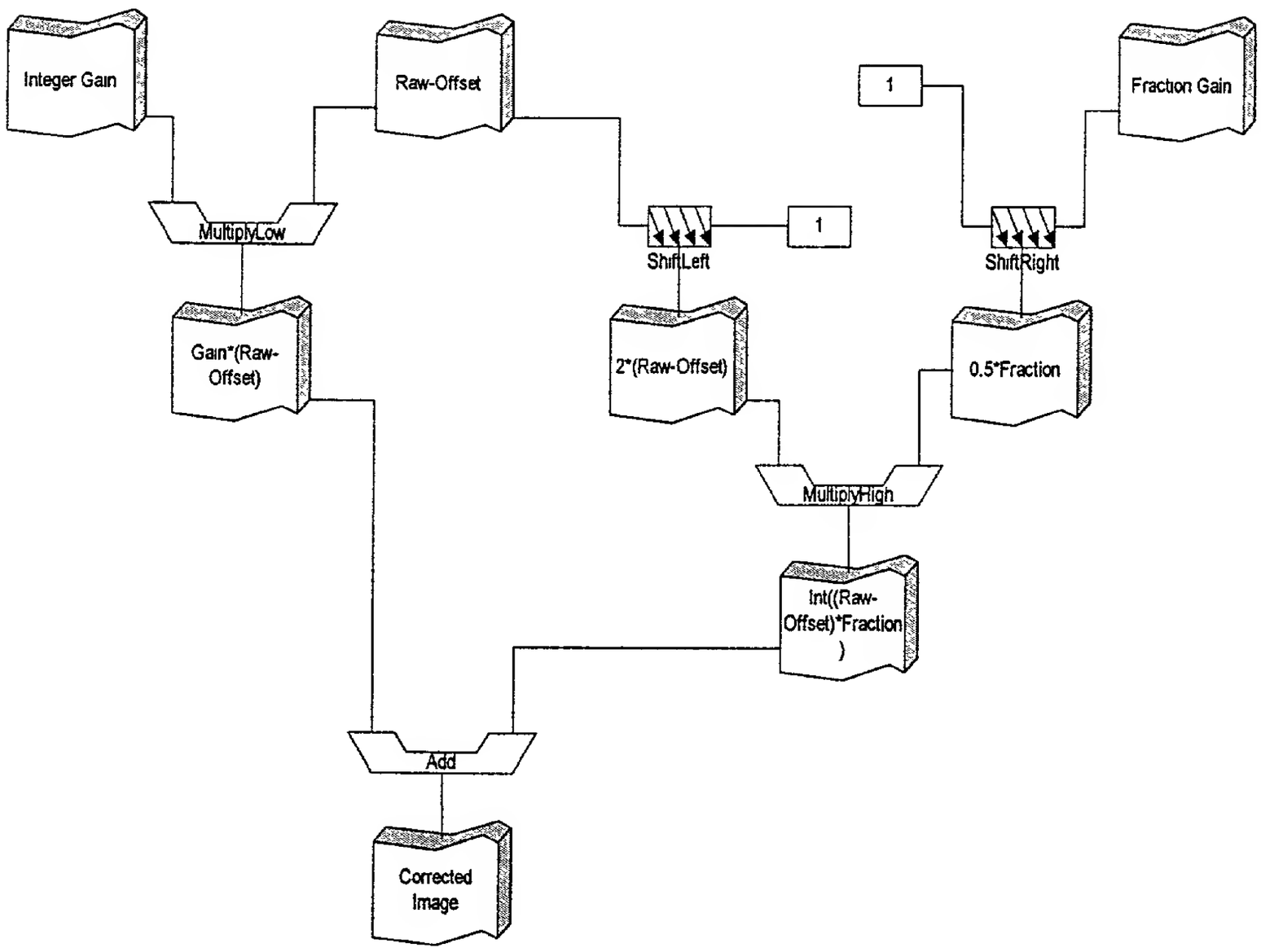
Integer Gain
Raw-Offset
1
Fraction Gain
MultiplyLow
ShiftLeft
1
ShiftRight
Gain*(Raw-Offset)
2*(Raw-Offset)
0.5*Fraction
MultiplyHigh
Int((Raw-Offset)*Fraction)
Add
Corrected Image

FIGURE 9

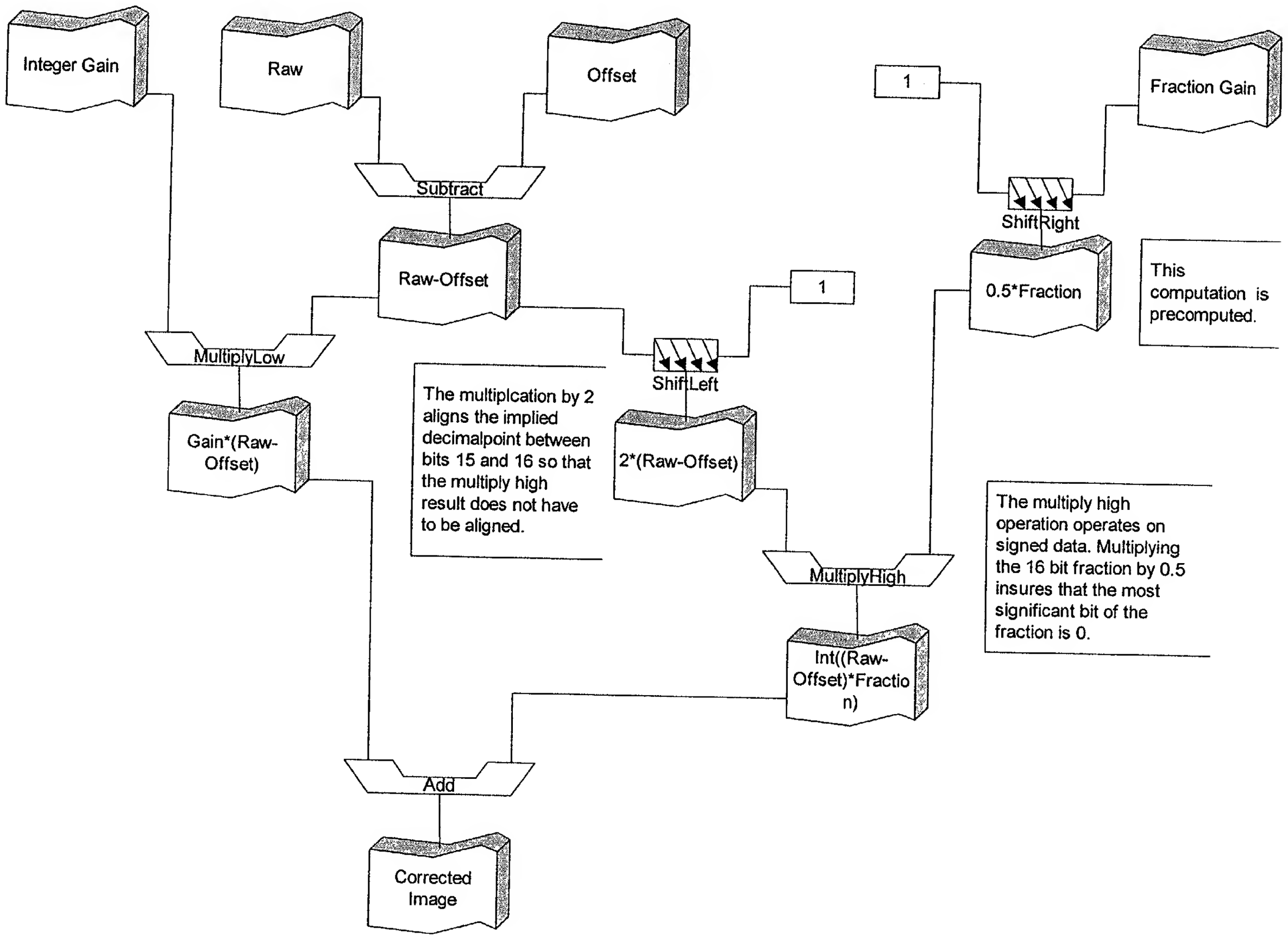
Integer Gain
Raw
Offset
Subtract
Raw-Offset
MultiplyLow
Gain*(Raw-Offset)
The multiplcation by 2 aligns the implied decimalpoint between bits 15 and 16 so that the multiply high result does not have to be aligned.
1
ShiftLeft
2*(Raw-Offset)
MultiplyHigh
Int((Raw-Offset)*Fraction)
1
Fraction Gain
ShiftRight
0.5*Fraction
This computation is precomputed.
The multiply high operation operates on signed data. Multiplying the 16 bit fraction by 0.5 insures that the most significant bit of the fraction is 0.
Add
Corrected Image

FIGURE 10

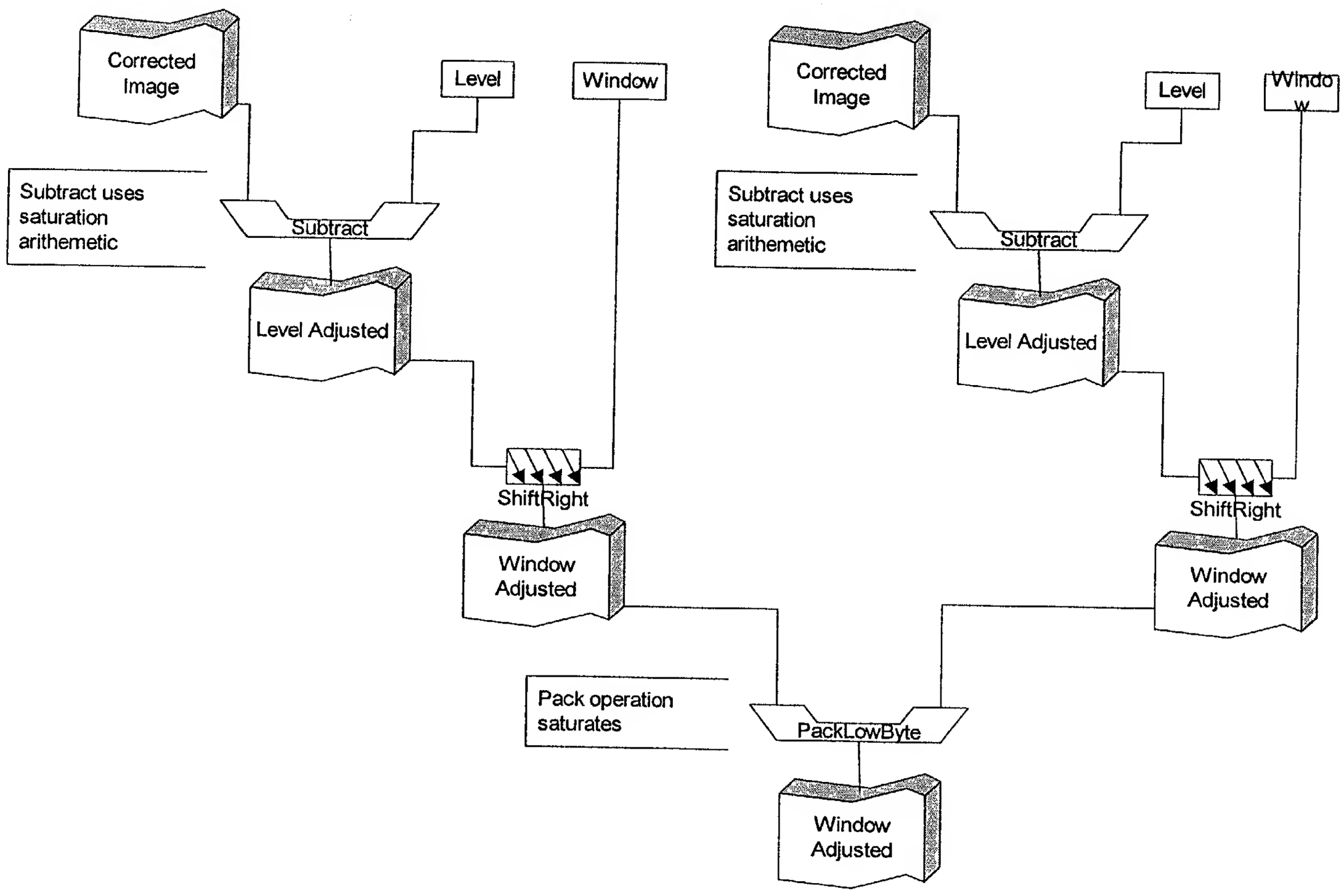
Corrected Image
Level
Window
Subtract uses saturation arithemetic
Subtract
Level Adjusted
ShiftRight
Window Adjusted
Pack operation saturates
PackLowByte
Window Adjusted
Corrected Image
Level
Windo w
Subtract uses saturation arithemetic
Subtract
Level Adjusted
ShiftRight
Window Adjusted

FIGURE 11

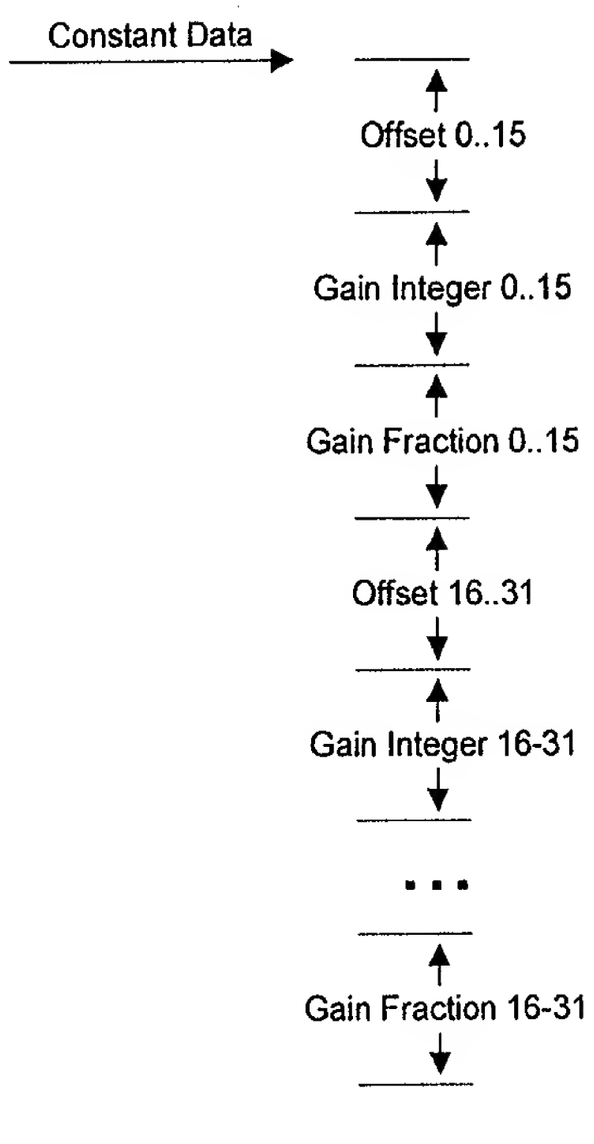
Constant Data
Offset 0..15
Gain Integer 0..15
Gain Fraction 0..15
Offset 16..31
Gain Integer 16-31
...
Gain Fraction 16-31

FIGURE 12

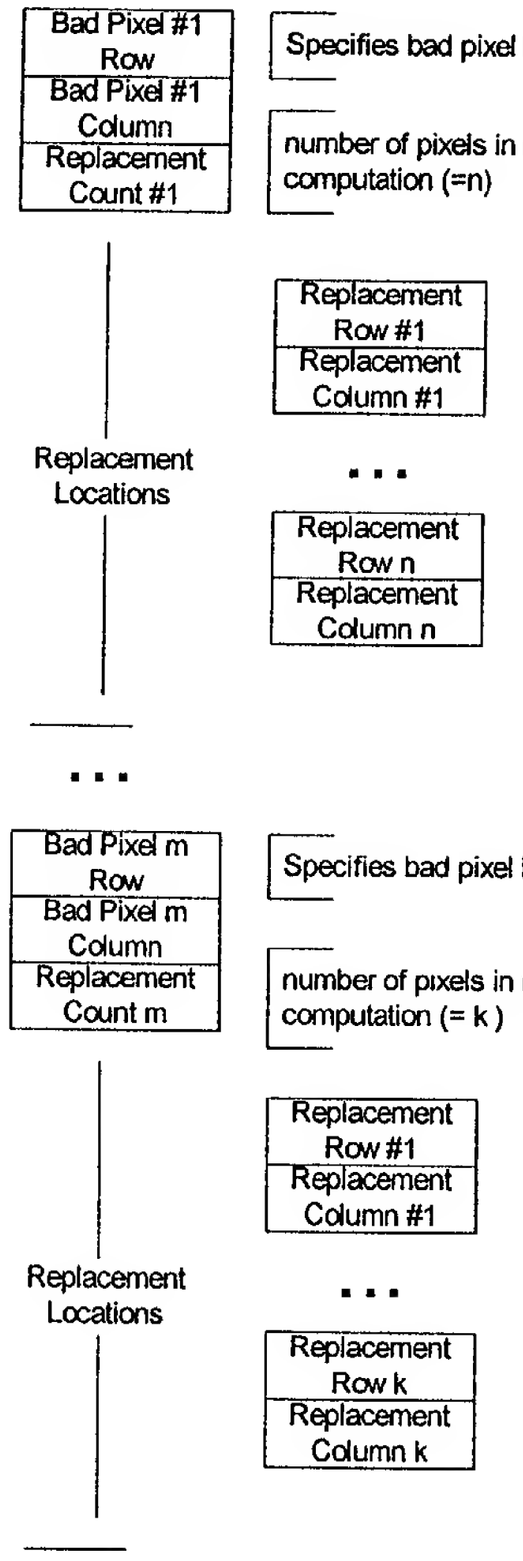
Bad Pixel #1 Row
Bad Pixel #1 Column
Replacement Count #1
Specifies bad pixel location
number of pixels in replace-ment computation (=n)
Replacement Row #1
Replacement Column #1
Replacement Locations
Replacement Row n
Replacement Column n
Bad Pixel m Row
Bad Pixel m Column
Replacement Count m
Specifies bad pixel location
number of pixels in replace-ment computation (= k)
Replacement Row #1
Replacement Column #1
Replacement Locations
Replacement Row k
Replacement Column k

FIGURE 13